M.T.S.
MULTIPROTOCOL TEST SUITE

TECHNICAL PRESENTATION

Fabien Henry
Fabien.henry@ericsson.com

MTS

ERICSSON

Table of contents

- Basics
- Input files
- Interfaces
- Protocol
- Core
- Logging
- Statistics
- Documentation
- Master/slaves
- Capture/Genscript
- Conclusion

MTS

Basics : features

ERICSSON

- Multi-protocol tester specially designed for IMS architecture
  - IMS protocols (SIP, RTP, DIAMETER, HTTP, GTP)
  - application ones (iptv, mail, sms, aaa, signaling, media, alarm/notif, nat, …)
  - basic transport ones (TCP, UDP, SCTP, TLS)

- Functional, non-regression or protocol tests => ‘Sequential’ mode
- Load, performance endurance stress tests => ‘Parallel’ mode
- Simulates equipments => client, server or both sides
- System supervision => capture mode (like wireshark)

- Definition of tests case in XML files : test and scenarios input files

MTS

Basics : features

ERICSSON

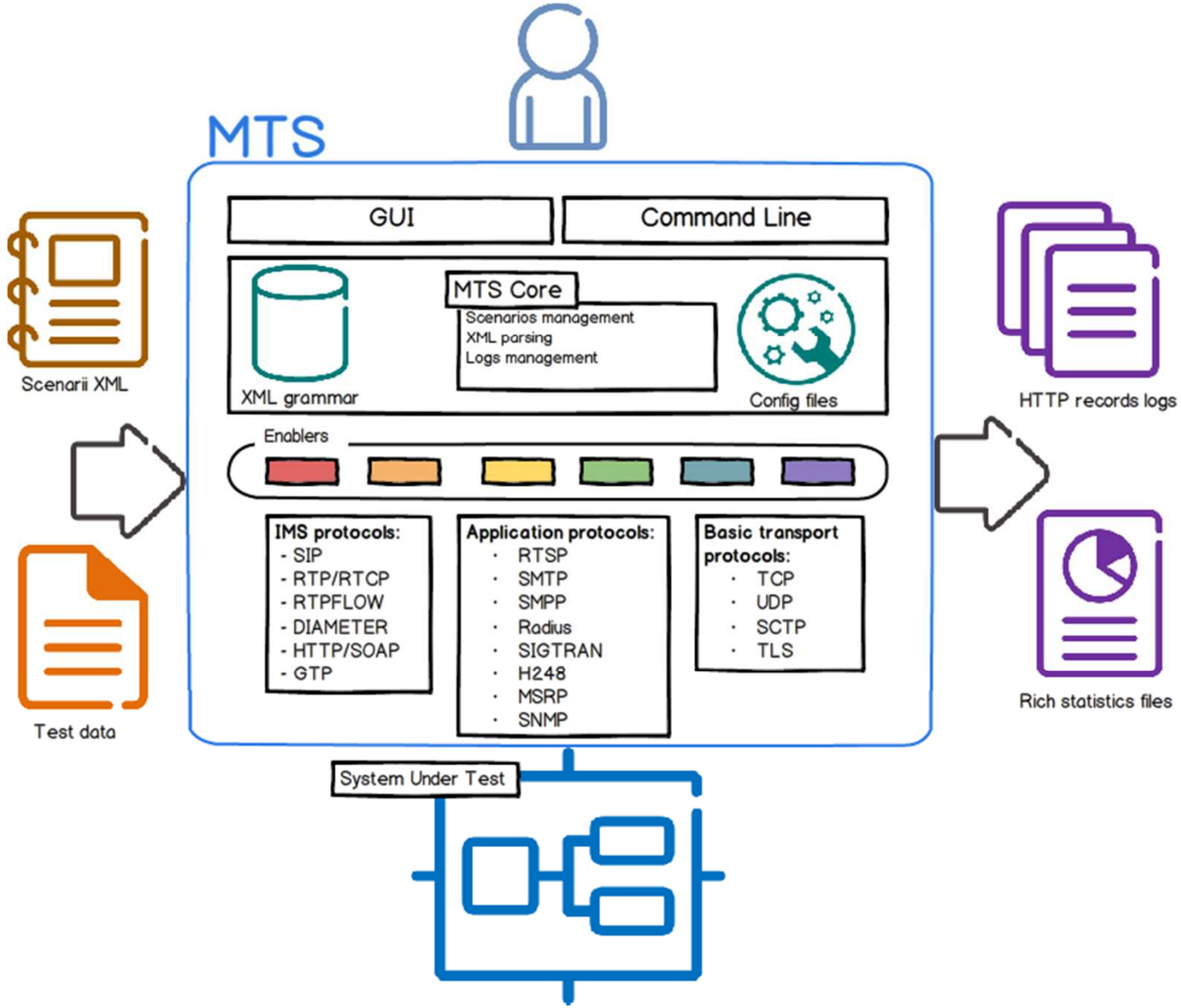
- Graphical (very convivial) or command line (for test automation) user interfaces
- Easy to use : logging management and rich statistics presentation.

- Pure software solution (use standard IP network card) => support only IP based protocol (for PSTN combined with Asterisk acting as a gateway)
- Written in java => supports many famous platforms : Windows and Linux supported
- Open Source product since begin 2012 with GPLV3 license => free to use

- Possible to run a test remotely with the master feature

MTS

Basics : architecture



MTS

Input files : test description

ERICSSON

- Described in XML syntax
- A test begins with a <test> tag → **tests suite**

- A test is defined as a testcase list
- Each testcase (<testcase> tag) is composed of a list of scenarios → **call-flow chart**

- For each scenario (<scenario> tag), the content is a reference to the XML scenario file which is relative to the test location or absolute → **call-flow entity**

- For all tags, you have “name” and “description” attributes
- **Changes in the test file need a test reload from the user**

- **XML syntax :**
  <test>/N*<testcase{state, number}>/N*<scenario>

MTS

Input files : scenario description

ERICSSON

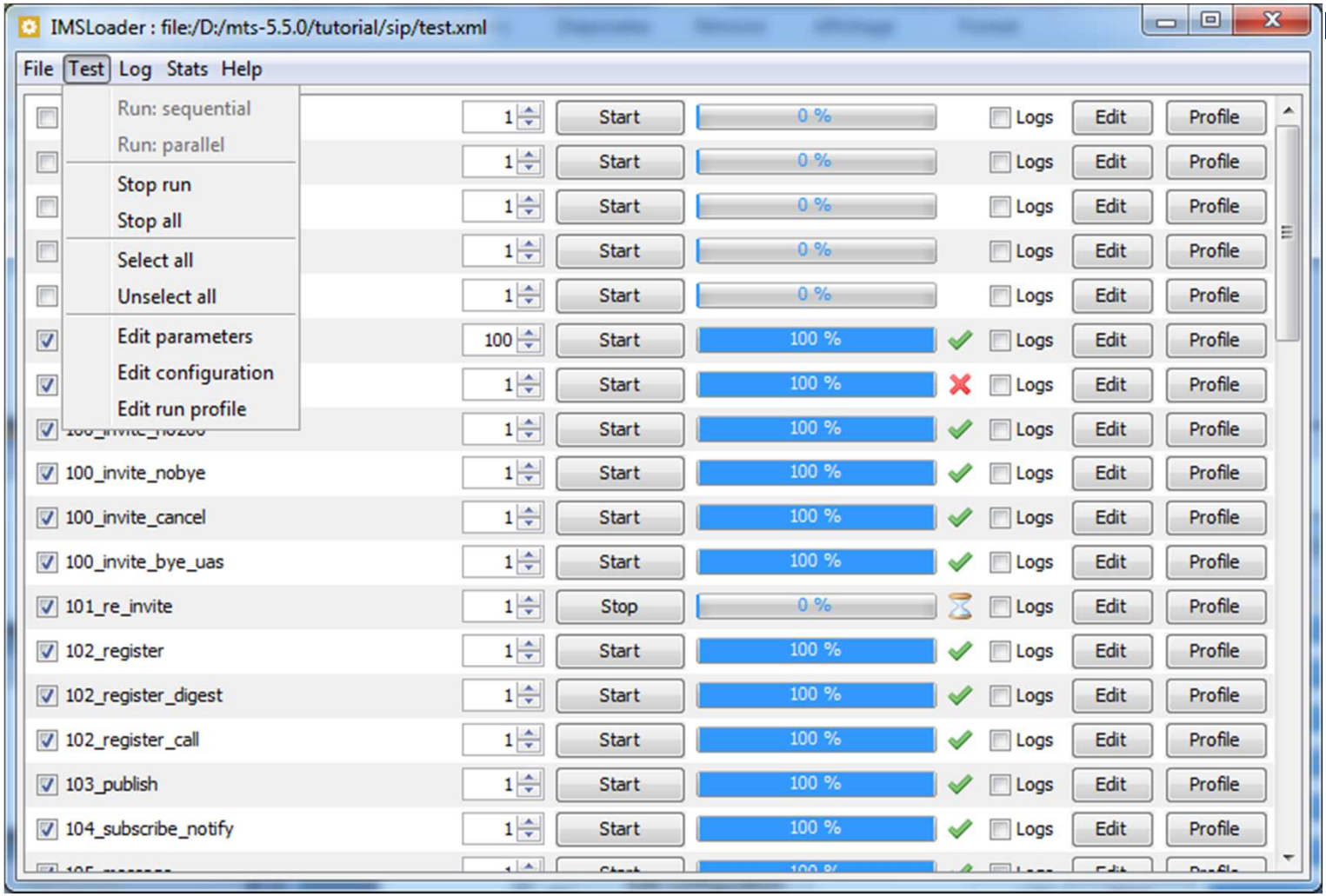
- Described in XML syntax
- A scenario file begins with the <scenario> tag
- Consists in a list of operations which are run sequentially in the time
- When an operation failed (often a timeout on message receipt), the scenario running is stopped and return a failure to the testcase
- Protocol operations (this is the main goal of the tool)
- Core operations
  - For all operations, you could defined 3 common attributes : “name”, “description” and “state”
  - **Changes in scenario file are taken into account immediately (without any reload)**
  - **XML syntax :**

    <scenario>/N*<operation>

MTS

Interfaces : GUI overview

ICSSON



MTS

Interfaces : GUI features

ERICSSON

- Useful for scripts writing, testing and debugging
- Starts with “All programs/mts-x.y.z” windows menu
- Go into the {root_dir}/bin directory and launch by the “startGui.xxx <testFile>” command
- Open, close, reload, edit the test file => menu “File”

  Show the list of testcases in the main windows

- Run a testcase manually one or many times => “Run” button
- Run a part of a test in sequential mode (only selected ones)
- Run a part of a test in parallel mode (simultaneously) at a given load (testcase by sec) => menu “Test”
- Configure the logging (file or memory) and display the memory main logs => menu “Log”
- Generate and show the statistic HTML report =>menu “Stats”

MTS

Protocol : operations → Standardized XML grammar (easy to learn)

- Based on message elements : data message format (text / binary)
  * **<sendMessagePPP>** => to send a message
  * **<receiveMessagePPP>** => to wait until a message is received with filtering features

- Uses a way of transport : a channel element (connection)
  * **<openChannelPPP>** => to open a transport channel
  * **<closeChannelPPP>** => to close a transport channel

- Listens dynamically to a host/port : listenpoint element
  * **<createListenpointPPP>** => to create a listenpoint
  * **<removeListenpointPPP>** => to remove a listenpoint

- Capture network traffic on a interface : probe element

  <createProbePPP> => to create a capture probe

  <removeProbePPP> => to remove a capture probe

MTS

Protocol : generic stack features

- Generic stack features available for all protocols
- Generic information on messages like “type”, “result”, “request” flag, transport information (listenpoint channel) ...
- Support of transactional protocol : transaction = a request and many response messages
- Manages automatically the retransmission of requests if no responses after a timeout (useful for un reliable transport as UDP)
- Perform retransmission filtering : not dispatch to the scenario
- Internal routing mechanism to dispatch a received message to the right scenario : using scenario name, message id, transaction id (for response)
- Support of session protocol : session = an initial messages and many subsequent requests

MTS

Protocol : supported main protocols (1/2)

Protocol	Stack	Transport	IO block	Trans action	Session	Listen point	Channel	Capture / Genscript
IMS / EPS								
SIP jain	Jain SIP stack	UDP,TCP, SCTP,TLS,RFC	N	Yes	Yes	Yes	No	No
SIP light	Java.net	UDP,TCP, SCTP,TLS,RFC	B/N	Yes	Yes	Yes	No	Yes/Yes UDP only
DIAMETER	DK open source	TCP, SCTP	N	Yes	No	No	No	Yes/Yes
RTP jmf	JMF stack	UDP	N	No	No	No	Yes	No
RTP light	Java.net	UDP	B/N	No	No	Yes	No	Yes/Yes
RTP flow	Java.net	UDP	B/N	No	No	Yes	No	Yes
HTTP	Jakarta	TCP, TLS	B/N	Yes	No	No	Yes	No
GTP V2, V1, Prime	Java.net	TCP, SCTP	B/N	Yes	No	Yes	Yes	No
Transport								
UDP	Java.net	UDP	B/N	No	No	Yes	No	No
TCP	Java.net	TCP	B/N	No	No	Yes	Yes	No
SCTP	DK open source	SCTP (Linux only)	B	No	No	Yes	Yes	No
TLS	Java.net	TCP	B	No	No	Yes	Yes	No

MTS

Protocol : supported application protocols (2/2)

Protocol	Stack	Transport	IO block	Trans action	Session	Listen point	Channel	Capture / Genscript
Application								
RADIUS	GP open source	UDP	B	Yes	No	No	Yes	No
RTSP	Java.net	TCP,UDP	B/N	Yes	No	Yes	Yes	No
SMTP	Java.net	TCP	B/N	Yes	No	Yes	Yes	No
IMAP	Java.net	TCP	B/N	Yes	No	Yes	Yes	No
POP	Java.net	TCP	B/N	Yes	No	Yes	Yes	No
SMPP	Java.net	TCP	B/N	Yes	No	Yes	Yes	No
UCP	Java.net	TCP	B/N	Yes	No	Yes	Yes	No
SIGTRAN (1)	Java.net	TCP,SCTP	B/N	Yes	No	No	No	No
H248 (text)	Java.net	UDP, SCTP	B/N	Yes	No	Yes	No	No
MGCP	Java.net	UDP, SCTP	B/N	Yes	No	Yes	No	No
MSRP	Java.net	TCP, TLS	B/N	Yes	No	Yes	Yes	No
SNMP	Snmp4j Mibble	UDP	B/N	Yes	No	Yes	No	No
STUN	Java.net GP util	UDP	B/N	Yes	No	Yes	No	No
ETHERNET	libpcap + jpcap	IP	N	No	No	No	No	Yes

(1) supports only M3UA, BICC, ISUP, SCCP layers and not ASN1 ones (like CAP, MAP, TCAP, …)

MTS

Protocol : benchmark

ERICSSON

- Short summary of the main load figures on a Dell 5400 computer (2x4 core RAM 4gb):

- 1) SIP : 300 CAPS out + 300 CAPS in, 90000 calls (SIP only)
- 2) SIP/RTP: 1200 calls (SIP/RTP send only)
- 3) SIP UA: 90000 subscribers (for 3 GB of RAM)
- 4) Diameter: 5000 transaction/s
- 5) HTTP: 4400 transaction/s

- To have more information about performance figures, please look at the benchmark document.

Core : main operations

- <parameter> : set an internal parameter with a specific operation type; eg with a setFromMessage operation, you could analyze a received message
- <test> : perform a basic test with a given condition type and put the test in the failed status
- <and>/<or>/<not> : Define complex condition by using logical operator combining many <test> operations.
- <if> : make a conditional branching with a complex condition
- <switch> : make a conditional branching with equals condition
- <label>/<goto> : jump to a label in the scenario running
- <while> : perform a loop in the scenario running
- <system> : run an external system command

Core : main operations

- <pause> : perform a wait in the scenario running
- <semaphore> : synchronizes 2 scenarios with “wait” (blocking) and “notify” (releasing) actions. DEPRECATED : use <parameter operation=“system.semaphoreXXXX”/> instead of.
- <exit> : stops the scenario execution and returns optionally a failure
- <finally> : force the execution of a block even when there is a failure
- <try> : escape errors on a sequence of operations

MTS

Logging : features

ERICSSON

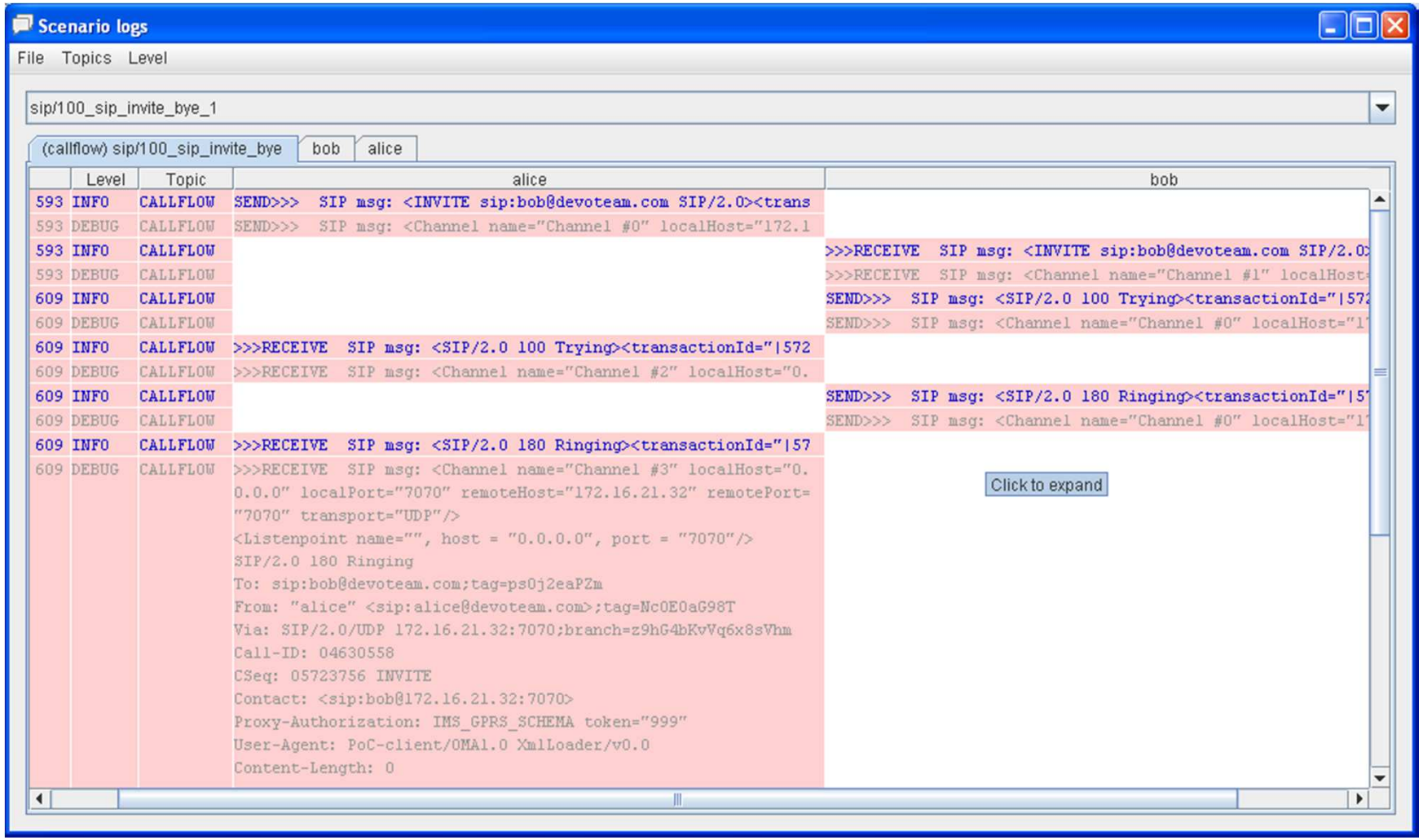
- Useful for functional, non-regression, protocol testing
- 2 scopes for logging messages: scenario and application
- 3 storage locations: on files or into memory (GUI only) or no location (logs disabled)
- 4 levels : the minimum level is configurable

  => DEBUG(0)/INFO(1)/WARN(2)/ERROR(3)
- 5 topics : for filtering

  =>CALLFLOW/PROTOCOL/CORE/PARAMETER/USER
- **<log>** operation : the user defines its own logging information

MTS

Logging : example



MTS

Statistics : features

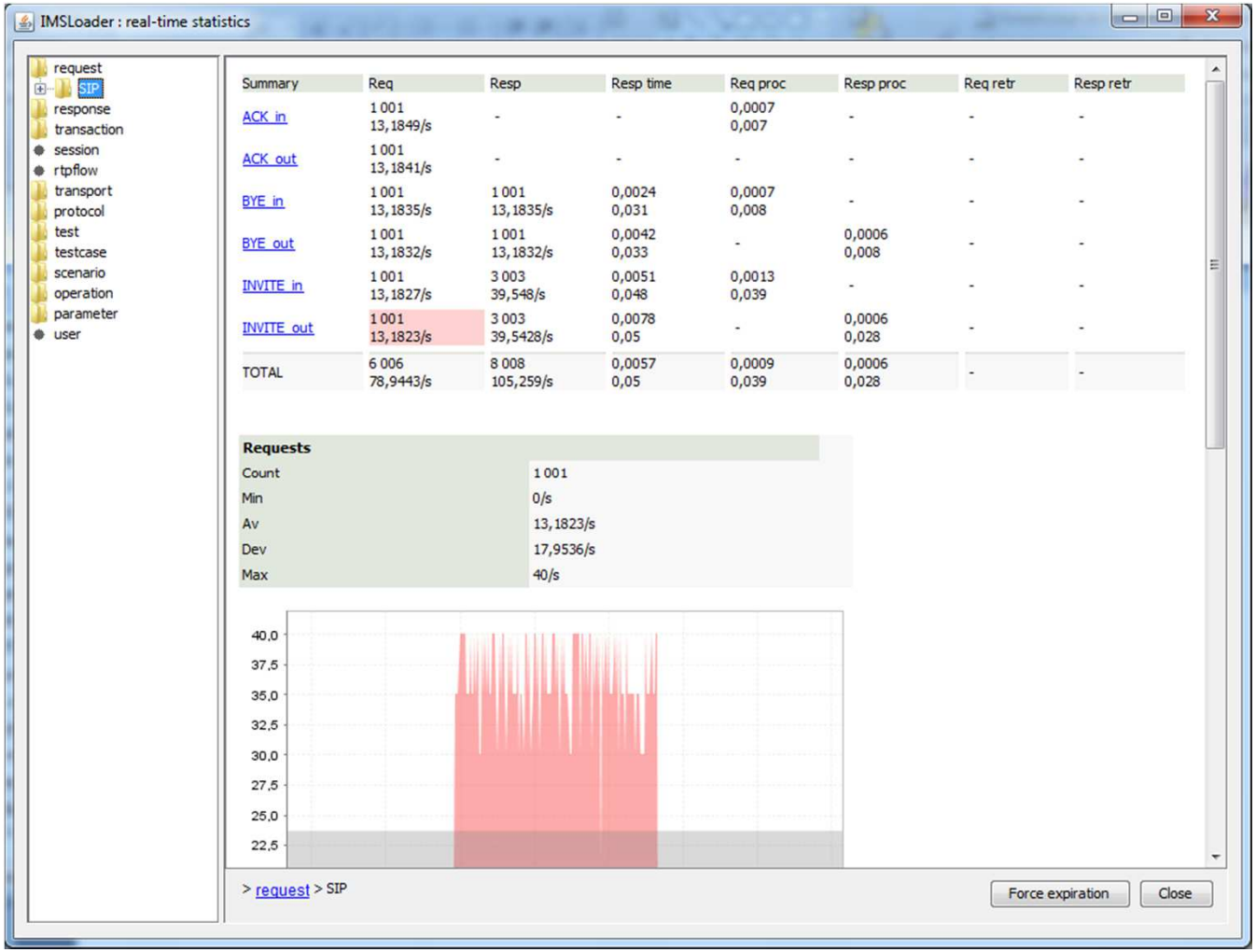
ERICSSON

- Used for load, endurance, stress testing
- A real-time statistics panel (GUI only)
- Generation of an HTML report statistics on user demand and automatically at the test end.
- Many predefined statistics counters organized in sections :
  =>Request/Response/Transaction/Session/RtpflowTransport/Protocol/
  Test/Testcase/Scenario/Operation/ Config/User
- **<stats>** operation : the user defines his own stat counter
- Counters organized as a tree to have more and more accurate information
- Temporal graph or value histogram or the same into the CSV format too

MTS

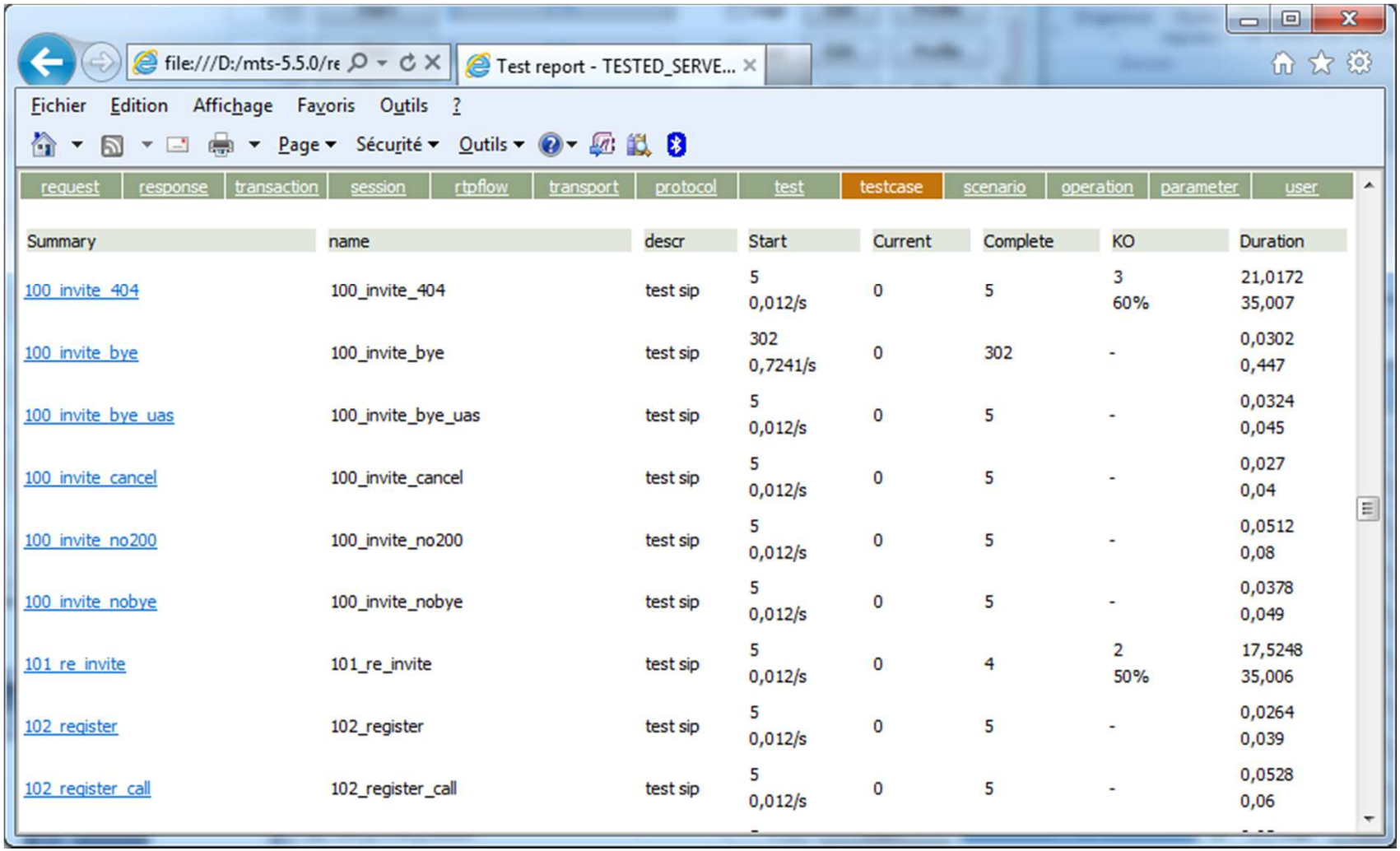
Statistics : real-time examples

ERICSSON



MTS

Statistics : HTML report examples



Summary	name	descr	Start	Current	Complete	KO	Duration
100_invite_404	100_invite_404	test sip	5 0,012/s	0	5	3 60%	21,0172 35,007
100_invite_bye	100_invite_bye	test sip	302 0,7241/s	0	302	-	0,0302 0,447
100_invite_bye_uas	100_invite_bye_uas	test sip	5 0,012/s	0	5	-	0,0324 0,045
100_invite_cancel	100_invite_cancel	test sip	5 0,012/s	0	5	-	0,027 0,04
100_invite_no200	100_invite_no200	test sip	5 0,012/s	0	5	-	0,0512 0,08
100_invite_nobye	100_invite_nobye	test sip	5 0,012/s	0	5	-	0,0378 0,049
101_re_invite	101_re_invite	test sip	5 0,012/s	0	4	2 50%	17,5248 35,006
102_register	102_register	test sip	5 0,012/s	0	5	-	0,0264 0,039
102_register_call	102_register_call	test sip	5 0,012/s	0	5	-	0,0528 0,06

MTS

Interfaces : command line (CMD)

ERICSSON

- Useful to run script automatically or with high performance
- Opens, runs the test and generates the statistics report
- Command startCmd.xxx usage :

  Usage: startCmd.xxx
  - <testFile> => the test description file (mandatory)
  - -seq[uential]|-par[allel]|<testcaseName> => the running mode (mandatory)
- Changes some editable parameters (-param option) or config values (-config) on the command-line
- Able to configure logging and statistic features with specific options on the command-line
- Some features are activated by key pressing (stop and generate report)

MTS

Documentation : tutorial & user manual

ERICSSON

- Tutorial : a complete a list of testcases given as examples

  * SIP call-flows (rfc 3261) and a lot of SIP extensions (INVITE, REGISTER, REGISTER_MD5, PRACK, UPDATE, PUBLISH, MESSAGE, SUBSCRIBE, NOTIFY, REFER, INFO, …)

  * SIP agent (PROXY, B2B, REGISTRAR, FORKING)

  * DIAMETER messages (rfc 3588) and all IMS interface (Cx, Dx, Rx, E2, E4, Rq, Gqp, …)

  * IMS Call-flow (TS 24228) (INVITE, INVITE_PRACK, REGISTER, REGISTER_MD5, SUBSCRIBE/NOTIFY, MESSAGE)

  * IMS core CSCF server (routing, authentication, AS trigerring features)

  => Learn the product or the protocols (pedagogic interest)

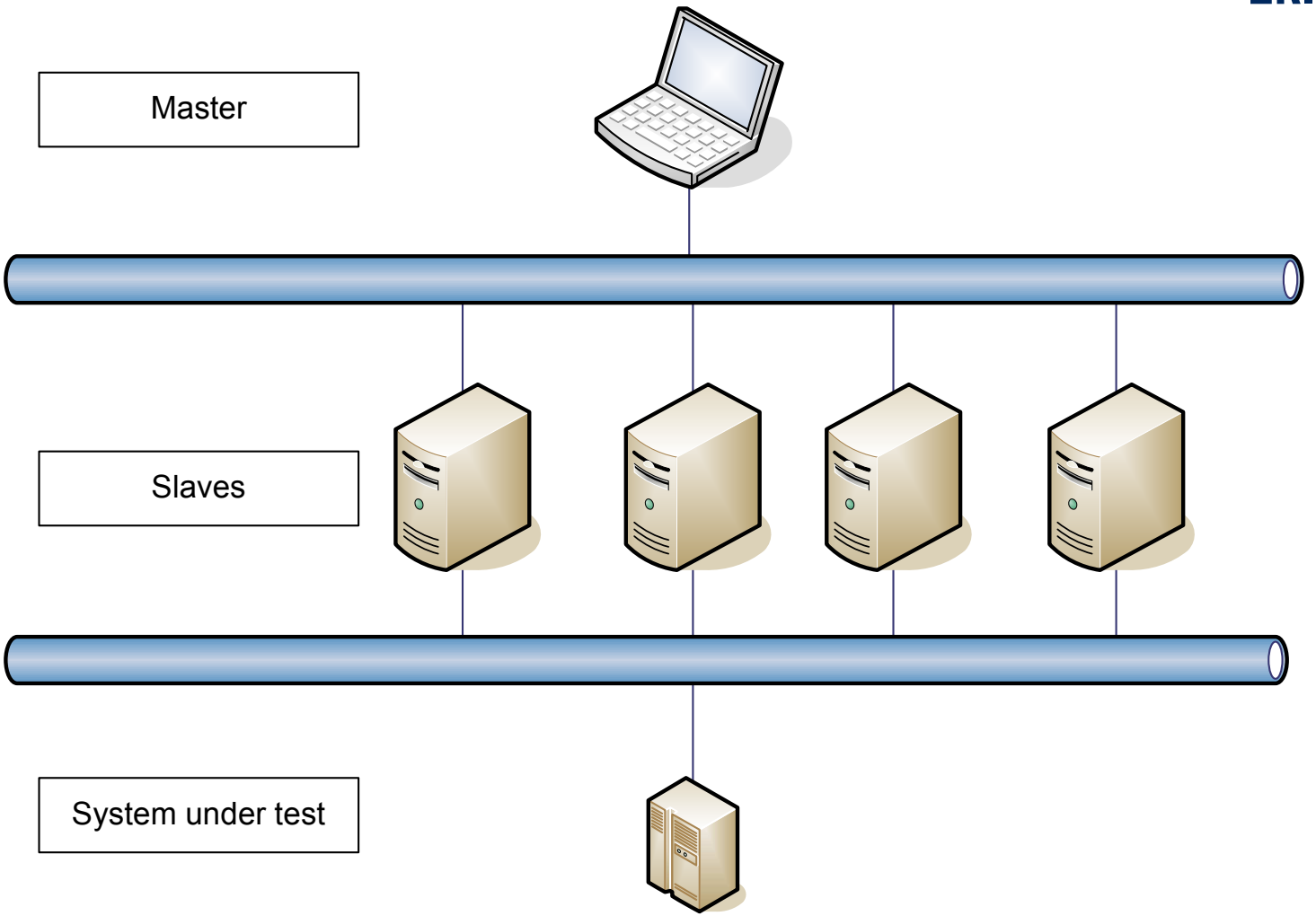
  => Start some new testcases as a starting point and just customize them (no blank page : => no worry for the tester)

- User manual : a detailed description of MTS features
- Technical presentation & demos : this one
- Provided with the product in the tool main directory

MTS

Master/slaves : architecture

ERICSSON



MTS

Master/slaves : features

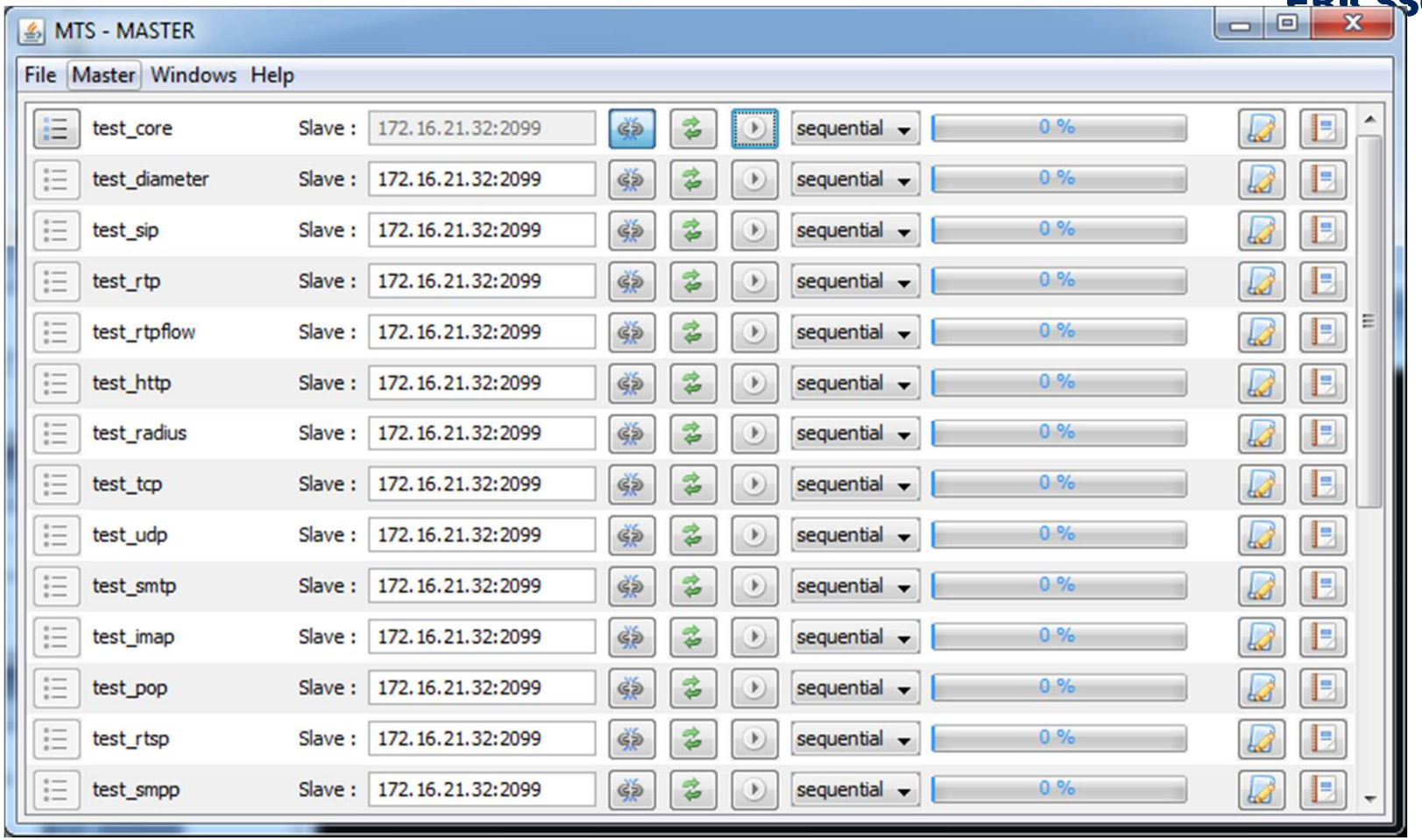
ERICSSON

- Master function : GUI user interface to launch and monitor the running of many tests on remote hosts
- Slave function : command line process to run a test on the host on master demand
- Communication between master and slaves uses RMI protocol
- Input => Master XML file with a list of tests and remote URL to reach the slave hosts
- Runs the selected tests sequentially or in simultaneously mode (parallel)
- Output => Tests progress and status, statistics HTML report

MTS

Master/slaves : GUI

ERICSSON



MTS

Capture mode

ERICSSON

- For supervision of production equipment
- No message generated by tool : complete passive behaviour
- Check the messages on the network, detect the bad one and react by statistic counter, logs information, mail sending or SNMP trapping
- Capture network traffic on a interface : probe element

  <createProbePPP> => to create a capture probe

  <removeProbePPP> => to remove a capture probe
- Of course all core operations available for getting information on a capture message (setFromMessage) and for testing it (<test> operation)
- Available on SIP (all transports), DIAMETER, RTP, RTPFLOW protocols
- Example provided in the tutorial for each of them

MTS

Genscript module

ERICSSON

- Generate automatically the XML scripts from a network capture (wireshark pcap format)
- Possibility to make a filter on messages contained into the capture file : based on a list of URL to simulate (<protocol>:<host>:<port> format)
- The user provides a capture file and a test file to get the result
- Create the scenario XML file (name is the port provided in URL)
- Generate <sendMessagePPP> and <receiveMessagePP> operations
- If port is not the default one, also generate <createListenpointPPP> and <removeListenpointPPP> operations
- Add the scenario into the test file at the testcase level (name is the capture file name); if testcase does not exist, then create it.
- Available on SIP (UDP only), DIAMETER, RTP, RTPFLOW protocols
- Example provided in the tutorial into <install_dir>/tutorial/genscript
- Usage: genScript.xxx
  - <URL list> => the list of URL to simulate (mandatory)
  - <captureFile> => the network capture file .pcap (mandatory)
  - <testFile> => the test description file .xml (mandatory)

MTS

Conclusion : benefits

ERICSSON

- Rich : supports of all IMS protocols) and transport ones
- Customizable : easy to change the call flows or messages
- Simple to use define the test; quickly run what you want to test
- Easy to test : No script to develop to reproduce a network capture using the genscript module
- Multi-purpose : functional, load testing, server-side simulation or supervision.
- Convivial : easy to use with its graphical user interface
- High-performance: about 5000 trans/s in DIAMETER and 400 CAPS in SIP
- Software : No specific hardware
- Universal : Run on most platforms
- Extensible : Easy development of a new protocol
- Scalable : provided by the master/slave features

MTS

Contacts

ERICSSON

CONTACT

Fabien HENRY

+ 33(0) 2 96 48 73 87

fabien.henry@ericsson.com

FRANCE

www.ercisson.com

Thank you for your attention

Aug, 13 2013

Authors	
Date of release	
File Info	